# CX-Supervisor
# Getting Started

**Software Release 2.0**

# Notice

OMRON products are manufactured for use according to proper procedures by a qualified operator and only for the purposes described in this manual.

The following conventions are used to indicate and classify precautions in this manual. Always heed the information provided in them. Failure to heed precautions can result in injury to people or damage to the product.

**DANGER!** Indicates information that, if not heeded, is likely to result in loss of life or serious injury.

**WARNING** Indicates information that, if not heeded, could possibly result in loss of life or serious injury.

**Caution** Indicates information that, if not heeded, could result in relatively serious or minor injury, damage to the product, or faulty operation.

# OMRON Product References

All OMRON products are capitalised in this manual. The word "Unit" is also capitalised when it refers to an OMRON product, regardless of whether or not it appears in the proper name of the product.

The abbreviation "PLC" means Programmable Logic Controller and is not used as an abbreviation for anything else.

# *Visual Aids*

The following headings appear in the left column of the manual to help you locate different types of information.

| | |
|---|---|
| **Note:** | Indicates information of particular interest for efficient and convenient operation of the product. |
| ***1, 2, 3…*** | Indicates lists of one sort or another, such as procedures, checklists etc. |
| | Represents a shortcut on the Toolbar to one of the options available on the menu of the same window. |
| | Indicates a program must be started, usually by clicking the appropriate option under the standard Windows 'Start' button. |

**© OMRON, 2007**

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form, or by any means, mechanical, electronic, photocopying, recording, or otherwise, without the prior written permission of OMRON.

All copyright and trademarks acknowledged.

No patent liability is assumed with respect to the use of the information contained herein. Moreover, because OMRON is constantly striving to improve its high-quality products, the information contained in this manual is subject to change without notice. Every precaution has been taken in the preparation of this manual. Nevertheless, OMRON assumes no responsibility for errors or omissions. Neither is any liability assumed for damages resulting from the use of the information contained in this publication.

# *About this Manual*

This manual describes the CX-Supervisor application and its installation.

This manual contains the following:

***Getting Started***. A description of CX-Supervisor installation and its use for the first time.

***Basic Tutorial***. A quick guide of use for the first time.

***Advanced Tutorial***. A detailed step by step tutorial for the key areas.

| | |
|---|---|
| **Warning:** | Failure to read and understand the information provided in this manual may result in personal injury or death, damage to the product, or product failure. Please read each section in its entirety and be sure you understand the information provided in the section and related sections before attempting any of the procedures or operations given. |

# TABLE OF CONTENTS

CX-Supervisor ..... Page

**Getting Started with CX-Supervisor ..... 1**

Welcome to CX-Supervisor ..... 1
About this Manual ..... 2
System Requirements ..... 3
Installing CX-Supervisor ..... 4
Copy Protection ..... 4
Starting CX-Supervisor ..... 8
Customising CX-Supervisor Settings ..... 9
The CX-Supervisor Help System ..... 9

**Basic Tutorial ..... 11**

**Advanced Tutorial ..... 25**

**Glossary of Terms ..... 49**

**Index ..... 53**

# Getting Started with CX-Supervisor

This chapter introduces the CX-Supervisor application to a new user.

## Welcome to CX-Supervisor

CX-Supervisor is dedicated to the design and operation of PC based visualisation and machine control. It is not only simple to use for small supervisory and control tasks, but it also offers a wealth of power for the design of the most sophisticated applications. CX-Supervisor boasts powerful functions for a wide range of PC based HMI requirements. Simple applications can be created rapidly with the aid of a large number of predefined functions and libraries, and even very complex applications can be generated with a powerful programming language or VBScript. CX-Supervisor has an extremely simple, intuitive handling and high user friendliness. Importing ActiveX® components makes it possible to create flexible applications and extend functionality.

CX-Supervisor offers a comprehensive range of facilities for the developer and is capable of developing solutions with the following features:

- Operator interface to processes
- Data acquisition and monitoring
- Information management
- Manufacturing control
- Supervisory control
- Batch sequencing
- Continuous process control
- Alarm monitoring and reporting
- Material handling (monitoring and control)
- Simulation and modelling through graphic animation
- Data Logging
- Error logging
- Project Editor and cross referencing
- Multi-lingual user interfaces
- Report editor
- Database Connectivity
- Connection to OPC Servers
- Use of ActiveX objects

- Use of Visual Basic script and Java Script

CX-Supervisor runs on standard PC desktop computers running Microsoft Windows. CX-Supervisor is intuitive and easy to use, and allows the developer to rapidly configure, test and debug a project. CX-Supervisor comprises two separate executable Windows programs, CX-Supervisor Development environment and CX-Supervisor Runtime environment. Applications are created and tested using the development environment and then delivered as a final customer application with the runtime environment.

The runtime-only environment may only be used for executing an application previously generated using the development environment. It is not possible to generate a new runtime application using the runtime environment.

**Note:** It is important that this copy of the CX-Supervisor software is registered on the OMRON web site, in order to qualify for technical support and maintenance updates. OMRON will not be able to help unless this copy has been registered.

## About this Manual

This manual helps a new user get started with CX-Supervisor, by describing the software installation and computer configuration, and by leading the user through the basics of CX-Supervisor application programming.

Separate OMRON manuals describe the related software products; CX-Server, CX-Programmer etc.

Some small example applications are included with the CX-Supervisor software to demonstrate some of the most useful features. These can be used to help with product familiarity.

CX-Supervisor comes with a comprehensive context-sensitive on-line help system, which is designed to complement this manual, and provide a quick reference at any point in the CX-Supervisor application when the manual is not to hand. This general help system uses a fast 'hypertext' system that allows progressively more information to be obtained about any topic by selecting keywords within the descriptive text.

Throughout this manual, it is assumed that the user has a working knowledge of Microsoft Windows, and can:

- Use the keyboard and mouse.
- Select options from Windows menus.
- Operate dialog boxes.
- Locate, open and save data files.
- Edit, cut and paste text.
- Drag and drop.
- Start programs from the “START” button.

If Windows has not been used before, spending some time working with the Microsoft documentation is recommended before using CX-Supervisor.

This introductory chapter deals with several important aspects of installing CX-Supervisor and setting it up for use. It is recommended that this entire chapter be read before installing the software.

# System Requirements

CX-Supervisor operates on the following hardware and operating systems:

## Hardware Requirements

The following configuration is the **minimum** system requirements for running CX-Supervisor:

- IBM PC compatible 600 MHz Pentium II processor or better.
- 256Mbyte of RAM minimum.
- 650Mbyte available hard disk space.
- 1024 x 768 XGA display.
- A CD ROM drive is required in order to install the CX-Supervisor software.

**Note:** These are requirements when using minimum operating system. For other operating systems consult your documentation.

The following configuration is the **recommended minimum** system environment for running CX-Supervisor effectively:

- IBM PC compatible Pentium II 1.0 GHz processor.
- 512Mbyte of RAM.
- 650Mbyte available hard disk space.
- 1024 x 768 XGA display.

## Operating Systems and Environments

The operating systems on which this software should be run are:

- Microsoft Windows 2000 (SP2)
- Microsoft Windows XP Professional

**Note:** This product does not install or run on Windows NT, Windows ME, Windows98 or Windows 95.

**Note:** CX-Supervisor is not guaranteed to be compatible with computers running Windows emulation (e.g. Apple Macintosh or VMWare).

**Note:** Internet Explorer 5.0 or later is required to use the version of CX-Server installed with this product

### Interfaces to Hardware Communications

CX-Supervisor utilises communications driver CX-Server for direct communication with OMRON factory automation equipment.

If it is intended for a PLC to be connected to the development computer for executing program code and testing, one of the following is required:

- RS-232C connection via a standard serial port on the computer (COM1 etc.)
- RS-422 connection to a 422 serial board
- Standard Ethernet board
- Standard USB port
- An OMRON Network Service Board

Refer to the appropriate hardware system manuals for full information about connecting and configuring these devices for the environment.

## Installing CX-Supervisor

The CX-Supervisor software is supplied on CD-ROM and is installed easily from within Windows. To install CX-Supervisor, load the disk in the CD-ROM drive. The AutoRun feature should automatically start the setup program and the installation of CX-Supervisor should take no more than a few minutes. If this has been disabled see the README.TXT on the CD-ROM for manual launching instructions.

After installation the text file README.HTML in the CX-Supervisor directory contains up-to-date information about the software. Read this file for details of any changes or new features in CX-Supervisor made since this manual was produced.

## Copy Protection

CX-Supervisor is copy protected to prevent illegal use. During installation a valid License Key must be entered. The CX-Supervisor Runtime environment has additional protection and if the protection is not installed certain features are inaccessible. To gain access to these features the USB protection must be installed.

### Copy Protection using a USB Dongle

The Runtime copy protection method consists of a small box called a “Dongle” which is plugged into a USB port. When CX-Supervisor is run the port is checked for a valid dongle and the program fails if it is not present.

### Fitting a CX-Supervisor USB Dongle

Simply connect the dongle to any spare USB port of the computer noting any instructions supplied. If there are no spare sockets a “USB Hub” device may be used to add extra sockets.

Once fitted, Windows will automatically detect the new hardware and the power light on the USB Dongle will illuminate.

**Note:** Old 'USB Dongles' from previous versions cannot be used with this version. The version 2.0 Runtime can only be activated with a new version 2.0 USB Dongle which are clearly identified as either ‘Machine Edition’ dongle or ‘PLUS’ dongle. A 'Machine Edition' dongle will only allow projects compiled for 'Machine Edition' to run. A 'PLUS' dongle will allow both 'Machine Edition' and 'PLUS' projects to run.

| Feature | Machine Edition | PLUS |
|---|---|---|
| ActiveX | Yes | Yes |
| VBScript | Yes | Yes |
| Recipes | Yes | Yes |
| Alarms | 300 | 5000 |
| Animations | Yes | Yes |
| Max no. Devices (PLCs etc.) | 15 | 256 |
| OPC Connections | Yes | Yes |
| Max user defined Points | 500 | 8000 |
| Max Regular Interval Scripts | 10 | 100 |
| Max no. Pages | 100 | 500 |
| Supported Databases | MS Access | SQL, ODBC, MS Access, MS Excel, dBase, CSV |

### Activating a CX-Supervisor USB Dongle

The USB Dongle should not require any additional activation. The CX-Supervisor installation process should correctly install and configure the required USB Dongle drivers, which should be automatically loaded when the USB Dongle is fitted as above.

In exceptional circumstances, the CX-Supervisor USB Dongle Drivers can be installed manually if required. The **USB Dongle Driver installation** shortcut can be started from the **Start** button, and can be found under the Omron and CX-Supervisor groups.

### Copy Protection using a Parallel Dongle

Old parallel port 'Hardlock Dongles' from previous versions cannot be used with this version. The version 2.0 Runtime can only be activated with a new USB Dongle..

### Copy Protection using a Software Token

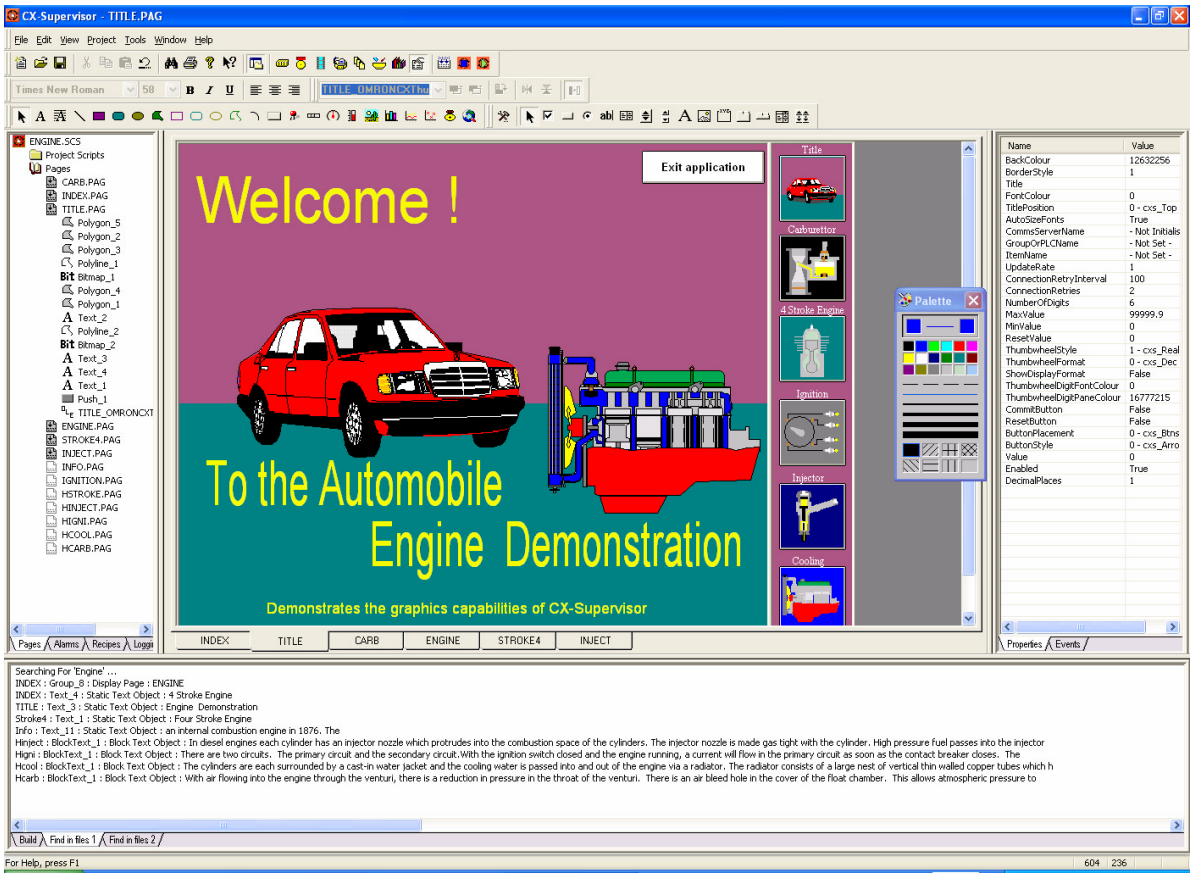
Old 'Software Tokens' from previous versions cannot be used with this version. The version 2.0 Runtime can only be activated with a new USB Dongle.

## Starting CX-Supervisor

After the software installation the *Programs* option from the **Start** button shows a new group window for CX-Supervisor under the OMRON heading. The software is ready to run and can be started by clicking on the CX-Supervisor Developer icon.

When CX-Supervisor Developer is started and a new Project created, it displays a window similar to the one shown below. When launched automatically from the installation, the Engine demo is automatically loaded, and by default when started thereafter the last loaded project is automatically loaded. The CX-Supervisor window offers many features to ease the process of programming, using the mouse or keyboard or both. It is possible to configure the display for any size monitor so that as much or as little information is visible as required, using options from the *View* menu. Any display configuration changes are saved in the Windows environment, and restored next time CX-Supervisor is run.

Use this section to become familiar with the layout of the CX-Supervisor display, and to set it up as desired.




## Customising CX-Supervisor Settings

By default CX-Supervisor shows the Workspace, Output view, Toolbar, Control bar and Status bar. The display of these can be selected by activating the options from the *View* menu.

To display a toolbar, Window or the palette toolbox, click on the menu item with the mouse. A tick next to a name indicates that it is currently displayed. CX-Supervisor saves the settings when it is exited and restores them when it is next executed.

## The CX-Supervisor Help System

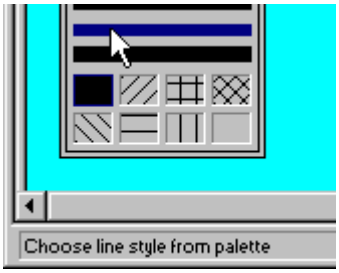
CX-Supervisor comes with a detailed context-sensitive help system: at any time while using the software, help can be obtained on the particular point currently being worked on, or on general

aspects of CX-Supervisor. This system is intended to complement the manual, by providing on-line reference to specific functions of the software and how to use them.

There are three types of CX-Supervisor help: index, context sensitive help and status bar help. An index of help topics can be accessed at any time by clicking *Index* from the *Help* menu. Instructions on how to use help can be accessed by clicking *Using Help* from the *Help* menu.

Alternatively, help on the topic currently in use can be obtained by pressing <F1>. Some dialogs also provide a **Help** button.

At the bottom of the CX-Supervisor screen is a general status bar, which provides several helpful pieces of information as the mouse is moved over display components.



CX-Supervisor also includes “point and click” help. To use this facility click on the icon. The cursor changes to an arrow with a question mark next to it. Use this cursor to point at any menu item or button on the display, and then click. Context-sensitive help for that item is then displayed.

General information regarding CX-Supervisor can be obtained by selecting *About CX-Supervisor* from the *Help* menu.

# Basic Tutorial

This chapter illustrates how to create a simple application using CX-Supervisor. It should only take about 20 minutes and should explain some key concepts:

- The difference between the Development and Runtime applications.
- The system is wholly controlled by data in the points, whether in memory or from a PLC
- Controls can set the data
- Graphical controls use Animations to control how the data is displayed

## Introduction

This chapter describes how a simple traffic signal is drawn using the Graphics Editor and how it is animated via the Animation Editor so that it changes colour – just as if it were a real traffic signal controlling a road or rail junction.

As the first tutorial proceeds, important concepts about CX-Supervisor are introduced by this symbol.

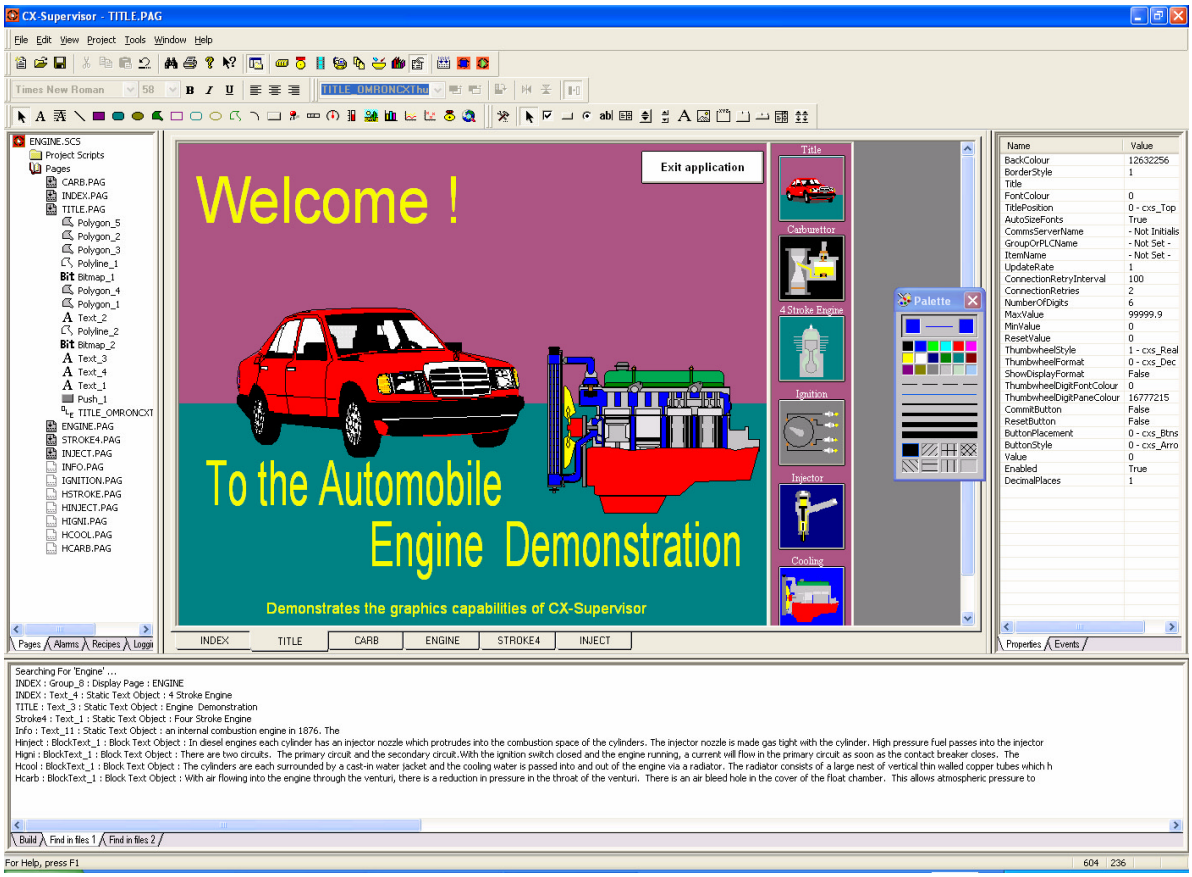
By the end of this tutorial, sufficient understanding of CX-Supervisor should have been gained to allow progression to the second, more advanced tutorial which demonstrates some of the more popular key features.

**Note:** CX-Supervisor uses standard Microsoft Windows dialogs and conventions wherever possible so that experienced Windows users should immediately feel comfortable. Inexperienced Windows users should refer to standard Windows documentation.

## The First Step

Before the tutorial can be started, the development version of CX-Supervisor must be installed on a suitable machine, as described earlier in this manual. When this has been done, invoke CX-Supervisor Development from the **Start** button.

When CX-Supervisor has initialised, the main screen is displayed as shown (or similar):



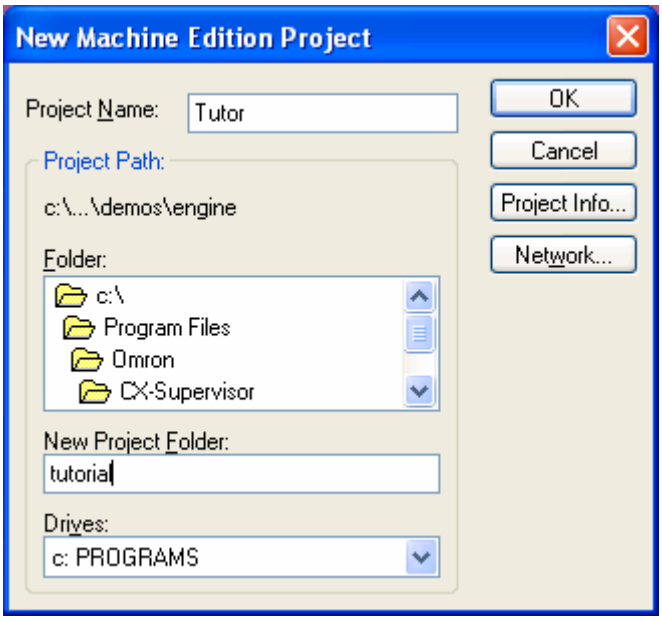

## The Concept of Projects

A project is the set of objects associated with one application. This includes Pages, Graphics, Reports, Alarms, Animations, Point configuration, Point values, Recipes and all other information.

Only one project can be opened by one application at a time.

A project must be created in which to conduct the tutorial; for simplicity, a separate sub-directory should be created for each project.

***1, 2, 3…*** Select *New Project* from the *File* menu and choose a Machine Edition project.




1. Enter the project name as “Tutor”.
2. Use the New Project Folder: field to determine the path to the new project directory (for example, there could be a directory path similar to “C:\Program Files\Omron\CX-Supervisor” on the machine).
3. Specify the name of the new sub-directory that CX-Supervisor is to create; for example, “tutorial”.
4. Click on the OK button. A project called “tutor” and the specified sub-directory is created by CX-Supervisor. The project file is actually called “tutor.scs”.

A blank page is displayed on the screen. If a blank page is not displayed, click on the **New Page** button from the toolbar or select *New Page* from the *File* menu to create a new, blank page.

## CX-Supervisor Pages

A CX-Supervisor project usually consists of a number of separate pages. Each page normally presents information relating to one particular topic, process, or activity. The application designer uses the facilities of the graphics editor to draw and animate objects on the page.

This tutor project consists of only one page.

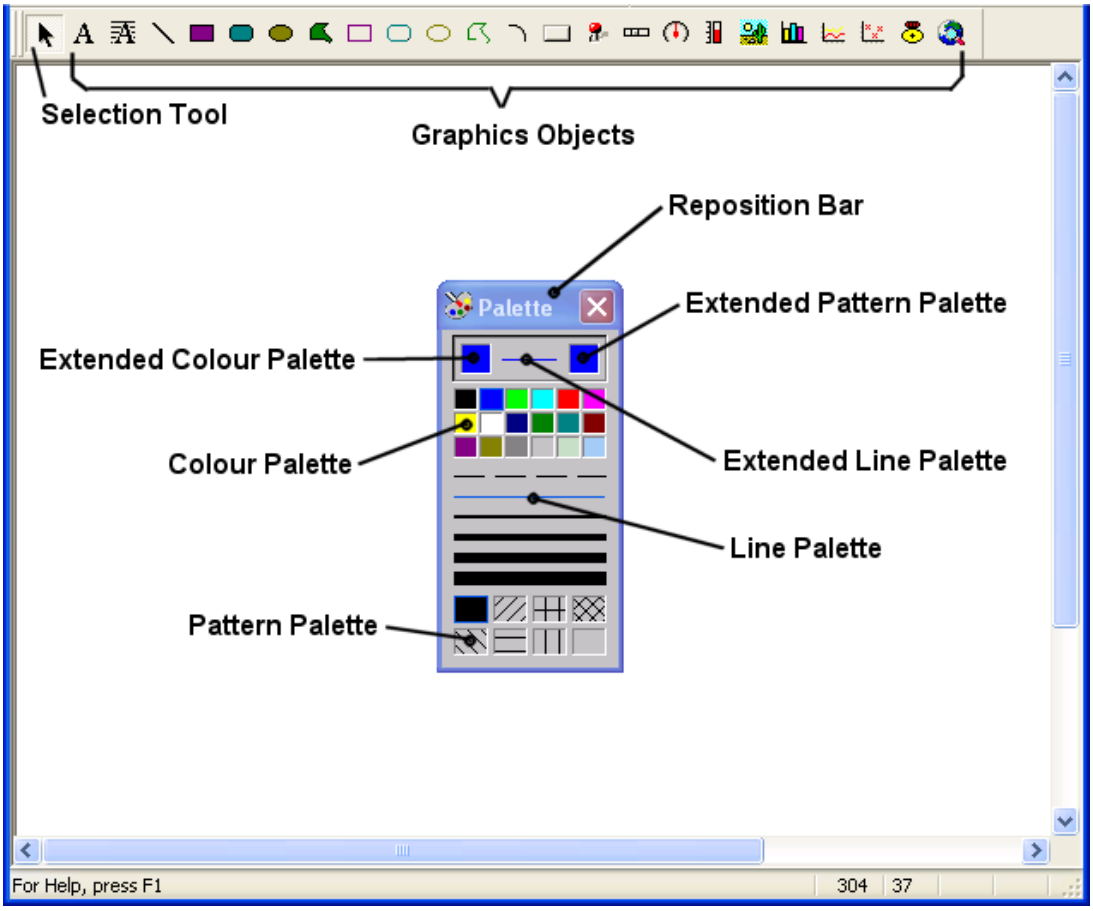
***1, 2, 3…***

1. Name the page as "tutor.pag" using *Save Page As* from the *File* menu.
2. Click on the **Yes** button when prompted to add the page to the project. The page displayed on screen should still be empty.

## Using the Graphics Editor

Now that the project exists with its own page, the graphic objects can be constructed and added to the page.

The graphics editor uses a Graphic Object toolbar and a floating window known as the Palette to construct and control objects on the page. These are very easy to use.




Several small icons are visible on the Graphic Object toolbar – each one representing one of the graphical objects with which an application can be constructed. Some of the objects are graphical

primitives – straight lines, ellipses, rectangles; some are rather more advanced – such as the gauge object, which has built-in functionality.

**Note:** The Graphic Object toolbar is only enabled when a page is active. If the toolbar looks disabled, click the page to make it active.

On the Palette is a colour palette, line-style palette and fill-pattern palette. These selections control the way in which objects appear.

**Note:** The Palette can be hidden and shown using the *View* menu.

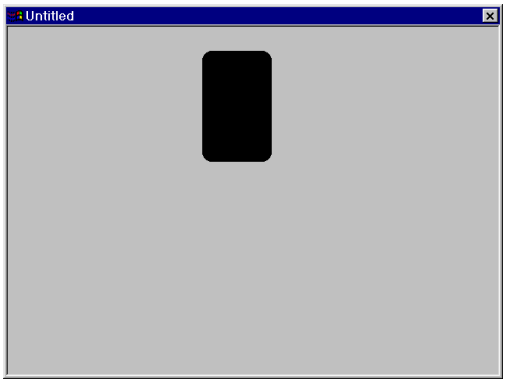
Experiment by creating examples of each of the various graphic objects at this stage and configure with the Palette. When finished with these objects, delete them by clicking on them and then hitting the <Delete> key or clicking on the **Cut** button from the toolbar.

***1, 2, 3…***

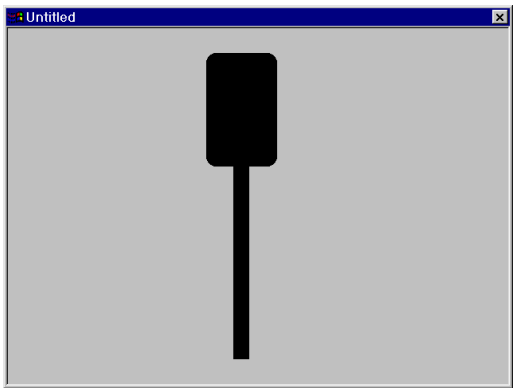
1. Select the **Select Mode** button from the Graphic Object toolbar. Notice that when the cursor is over an icon on the palette information about it is displayed on the status bar. Similarly, if the cursor is left stationary over a button then a tooltip message appears after a short while.

2. Ensure that the selected colour from the palette is black, by clicking on the colour black. Select the **Round-Rectangle** button from the Graphic Object toolbar.

3. Click the left mouse button and move the mouse pointer to near the top of the page and drag the pointer downward. As this happens, the outline of a round-rectangle appears on the screen to form the housing of the traffic signal as shown below. Notice that the position, height and width of the object being edited are displayed for reference on the status bar.



4. Select the **Rectangle** button from the Graphic Object toolbar and draw the support leg of the traffic signal.

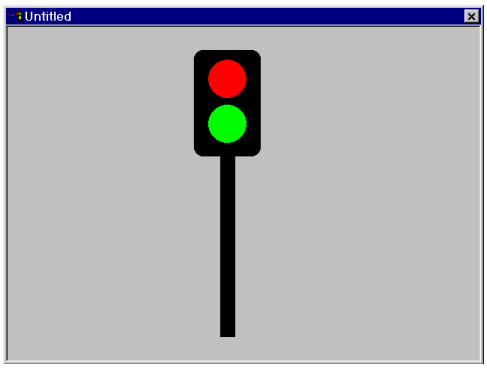


5. Now that the main structure of the traffic signal is complete (although rather basic), work can begin on the coloured lights. Only two lights are needed – one red and one green; this is best achieved by selecting the **Ellipse** button from the Graphic Object toolbar and drawing a circle of appropriate size to represent the light.

**Tip:** Holding the CTRL key down while drawing an ellipse ensures that a true circle is drawn.

6. Select the colour red and then position the light at the top of the traffic signal.

7. Repeat the process for the green light and position this at the bottom of the traffic signal.




## Adding a Point

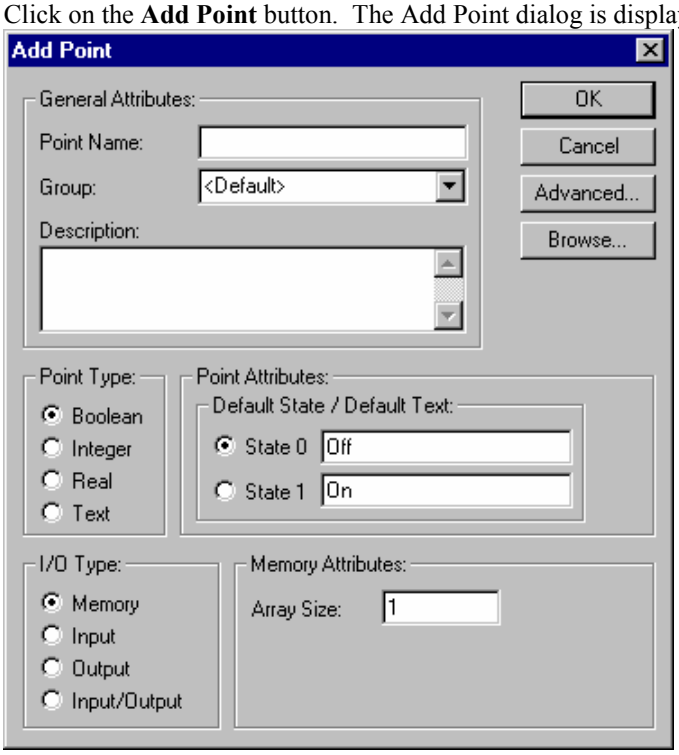
Now we need to define a variable by which the traffic signal is to be controlled. As there are only 2 states i.e. 'GO' and 'STOP' a Boolean or digital value will be used.

For simplicity of this tutorial this 'Point' or variable will be a local memory point. For a real application of course we would be connecting to a PLC and reading a digital input. Except for the following point definition the rest of this tutorial is *identical* to a real monitoring application.

***1, 2, 3…***

1. Click on the **Point Editor** button on the main toolbar. The Point Editor window is displayed.

2. Click on the **Add Point** button. The Add Point dialog is displayed.



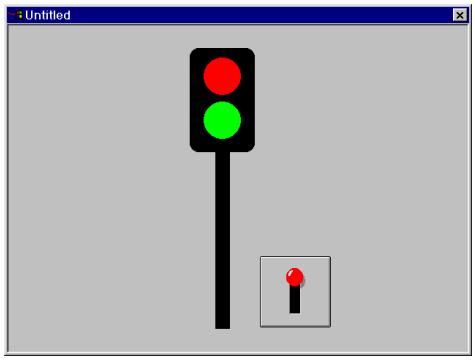

3. Now the variable, or 'Point', can be defined to control the traffic signal; in this instance a simple Boolean point is used.
4. At the *Point Name* field, type "GO", as the name of the point.
5. Check that the *Point Type* is "Boolean" and that "Memory" is selected as the *I/O Type*.
6. Click on the **OK** button. The point is added to the points database.

## Adding the toggle button

As this is a simulation to see the application in action we need a means of changing the value of our new point "GO". The simplest method is to use a "Toggle button", a two-state button which changes from one state to the other each time the button is pressed.

***1, 2, 3…***

1. Select the **Toggle Button** from the Graphic Object toolbar and draw out a button of a suitable size in relation to the traffic signal.




2. Double-click on the newly-created **Toggle button** to bring up the Toggle Button Wizard dialog.
3. We will link the toggle operation to our new point. Click on the **Browse** button and the Select Required Item dialog is displayed
4. Select the new point 'GO' and click on the **OK** button.
5. Click on the **OK** button on the Toggle Button Wizard dialog.

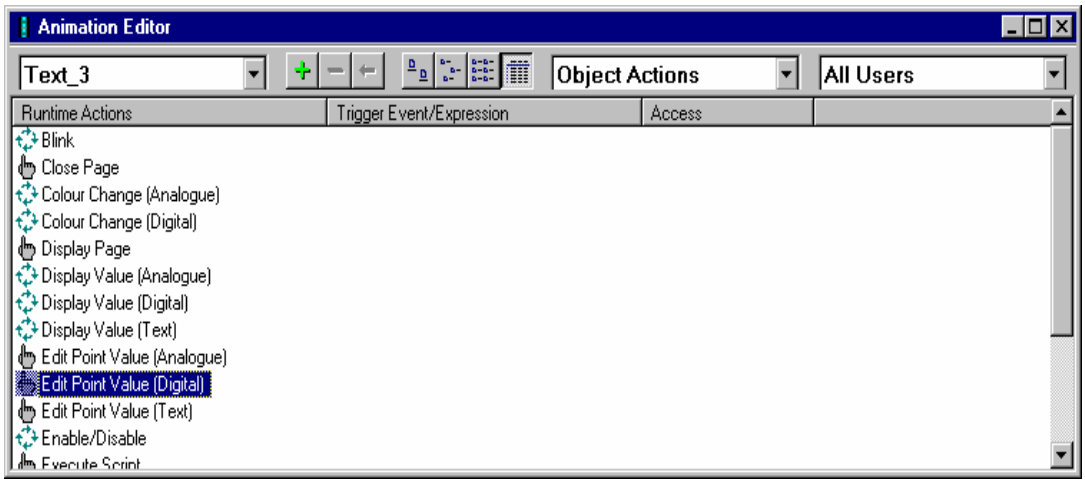
## Animation

This simple traffic signal has only two states, 'STOP' and 'GO'. If the signal is 'STOP', then the red light must illuminate. If the signal is 'GO' the green light must illuminate. The Animation Editor is the key to defining how an object is *animated* depending on point values. To configure the animations:

***1, 2, 3…***

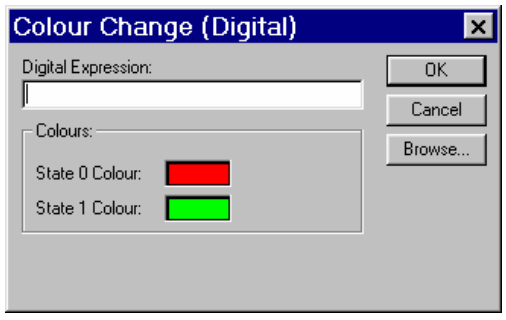
1. For simplicity first select the object that represents the **Green** light.

2. Click on the **Animation Editor** button from the toolbar.



3. There are a number of actions available for each particular object. With suitable configuration, each of these actions may be applied to each object; for example, the green light can be given the following actions:
   Blink.
   Close Page.
   Colour Change, etc.

4. By double-clicking the left mouse button, choose *Colour Change (Digital)*. By selecting this animation, CX-Supervisor is informed that the colour of the Green light (an ellipse) is to be changed according to the value of some Boolean (digital) variable or expression. The Colour Change dialog appears:



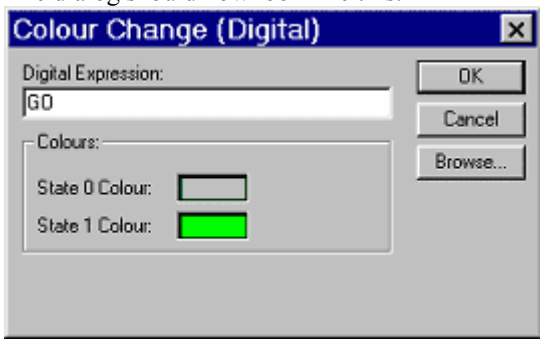

5. Now we will link this animation to our point. Click on the **Browse** button on the Colour Change dialog. The Select Required Item dialog is displayed.

6. Select the new point 'GO' and click on the **OK** button.

7. In the Colour Change (Digital) dialog, ensure that "GO" is entered as the digital expression.

8. The Colour Change dialog shows the two colours between which the light changes. The colours need to be selected so that when the traffic signal is 'GO', the Green light is bright-green (as per the default), but when it is 'STOP' i.e. State 0, the Green light dark-green or grey indicating the bulb is not illuminated. The colours can be changed by clicking on the colour preview box, and then using the Colour Palette dialog which subsequently appears to choose the appropriate colour tones.

9. The dialog should now look like this:




Click on the **OK** button.

The animation of the green light has now been defined.

The application should be tested to ensure that the light is operating correctly. Although the Red light could also be animated at this stage, it is probably worth performing a runtime test on the tutor application just to see how it operates.

## Testing the Project

Now the project can be tested in the runtime environment.

***1, 2, 3…***

1. Click on the **Run** button from the toolbar. CX-Supervisor first saves the current project and asks for confirmation to save the latest changes to "tutor.pag". Click on the **Yes** button.
2. If this is the first time the page is saved, you will be prompted to add it to the Project – select "Yes".
3. As there are no other pages to display, you will be prompted to Display the page on Startup, select "Yes".
4. If you are prompted to save the project "Tutor.SCS", select "Yes".
5. The runtime system starts with the page just created shown inside a frame window.
6. Click on the **Toggle** button and observe the button change state. The Green light should change colour. When the button is clicked again, the Green light should revert to its previous colour. Repeat this test a few more times to observe the application in action. Notice that it really doesn't matter how quickly or slowly the button is selected; the light can always keep up with the changes.

Be sure to check the colour of the Green light. Is it changing correctly between the two states? If not, check the colour definitions back in the development environment.

## Refining the Project

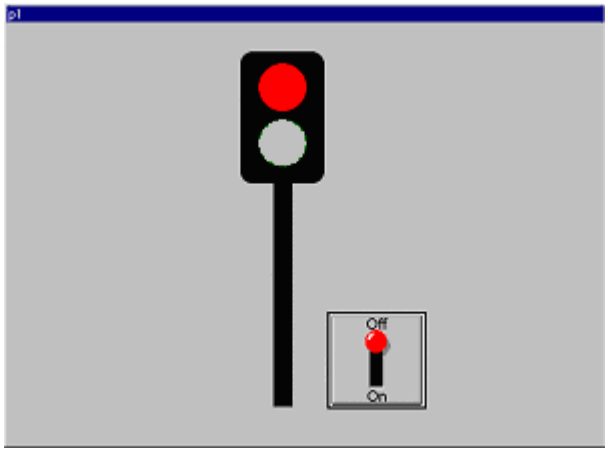
Any problems can be rectified by refining the project and re-running the application.

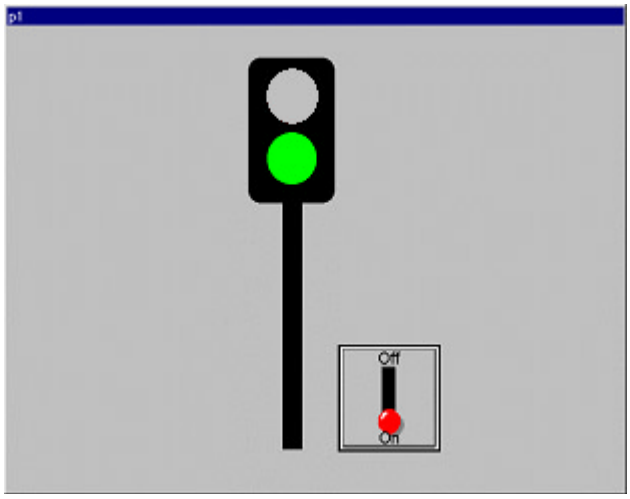
***1, 2, 3…***

1. Exit the Runtime environment by selecting *Close* from the *Control* menu, or Right Clicking and selecting *Exit* from the popup menu.
2. In the development environment, select the Green light and then bring up the Animation Editor dialog.
3. Once again, choose the Colour Change (Digital) dialog. The two colours are shown at the bottom of the dialog. If necessary, select the colour and change its tone by using the Colour Palette dialog.
4. Now select the Red light and animate this in the same manner as for the Green light. Remember that with the traffic signal, the red light needs to be 'ON' i.e. bright-red when 'GO' is false i.e. State 0. When 'GO' is true i.e. State 1 the Red light should be dark-red or grey indicating the bulb is not illuminated.

Now try runtime once again. Click on the **Run** button from the toolbar and try out the new tutor application; this time both lights should change colour as the button is selected. If both lights go on and off together then check, and possibly, change the colour definitions as described above.

Otherwise the traffic signal should be working as expected - changing from red to green as the button is pressed. On alternate selections, the signal should change back from green to red.





Admittedly this application is very simple, but it demonstrates some key points for all your applications:

- The difference between the Development and Runtime applications.
- The system is wholly controlled by data in the points, whether in memory or from a PLC
- Controls (like the Toggle button) can set the data
- Graphical controls use Animations to control how the data is displayed

This concludes the Basic Tutorial and you have learnt a little about Projects, Pages, Graphical objects, Points, Animations and running projects.

# Advanced Tutorial

This chapter introduces some of the more advanced topics of CX-Supervisor. It is expected the user is already familiar with:

- General Windows operation
- Software installation
- Creating, opening, saving and running of CX-Supervisor projects.
- Creating and saving pages
- Adding graphical objects and animations

This chapter is intended to be a quick guide to the topics, not full step-by-step instructions. For a full description of these areas refer to the related chapters in the User Manual.

## Coffee Machine scenario

This tutorial is going to take the form of a simulation of a coffee vending machine. They all have a familiar User Interface, display functions, control functions and we will be adding alarm, recipe and security functions. Again this tutorial will use Memory points for simplicity, but remember with minimal changes the application could easily be monitoring and controlling a real PLC.

## STEP 01 - GETTING STARTED

Follow the steps outlined, referring if necessary to the User Manual for specific instructions.

***1, 2, 3…***

1. Create a new Machine Edition Project with the following specifications:

   Project Name = **COFFEE**
   Project Description = **Coffee Vending Machine**
   Project Directory = **C:\COFFEE**

2. Create a new page from the File menu. Save the created page as "**Open.pag**" into the project directory, **C:\COFFEE**
3. Make sure the page is added to the project, if not add it
4. Run the application and test if the page is displayed correctly

## STEP 02 – PROJECT EDITOR

***1, 2, 3…***

1. Create a new page from the File menu
2. Save the new page and name it "**Main.pag**"
3. Using the Workspace, right click **Main.pag** to make it **Display on Run** and make **Open.pag** not **Display on Run** by removing the tick.

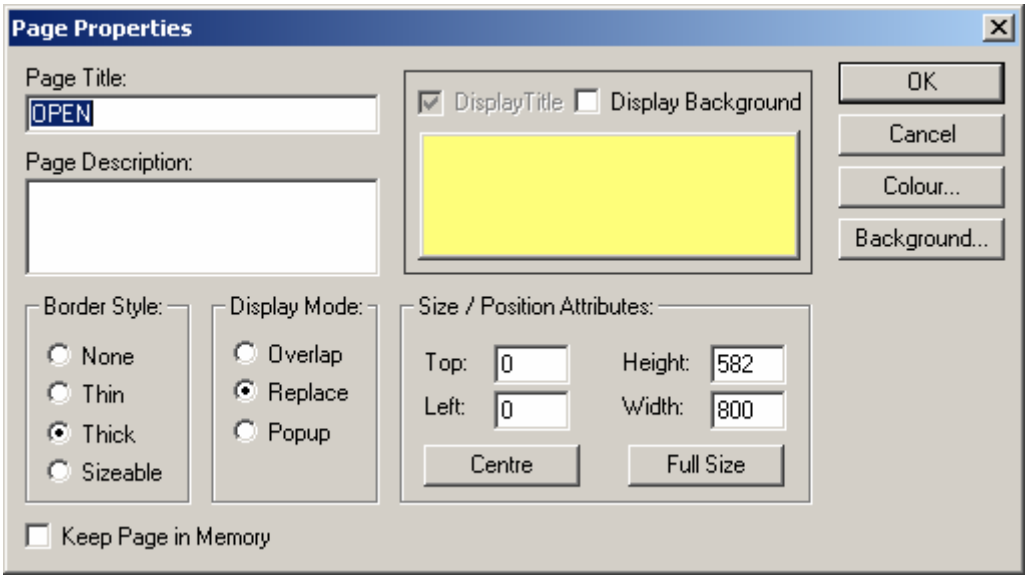
# STEP 03 – GRAPHICS PAGES

***1, 2, 3…***

1. Open **open.pag**
2. Double click the page to set the following properties:

   Page Title = **Open**
   Border Style = **THICK**
   Display Mode = **REPLACE**
   Top = **0**
   Left = **0**
   Height = **582**
   Width = **800**
   Background colour = **Yellow**

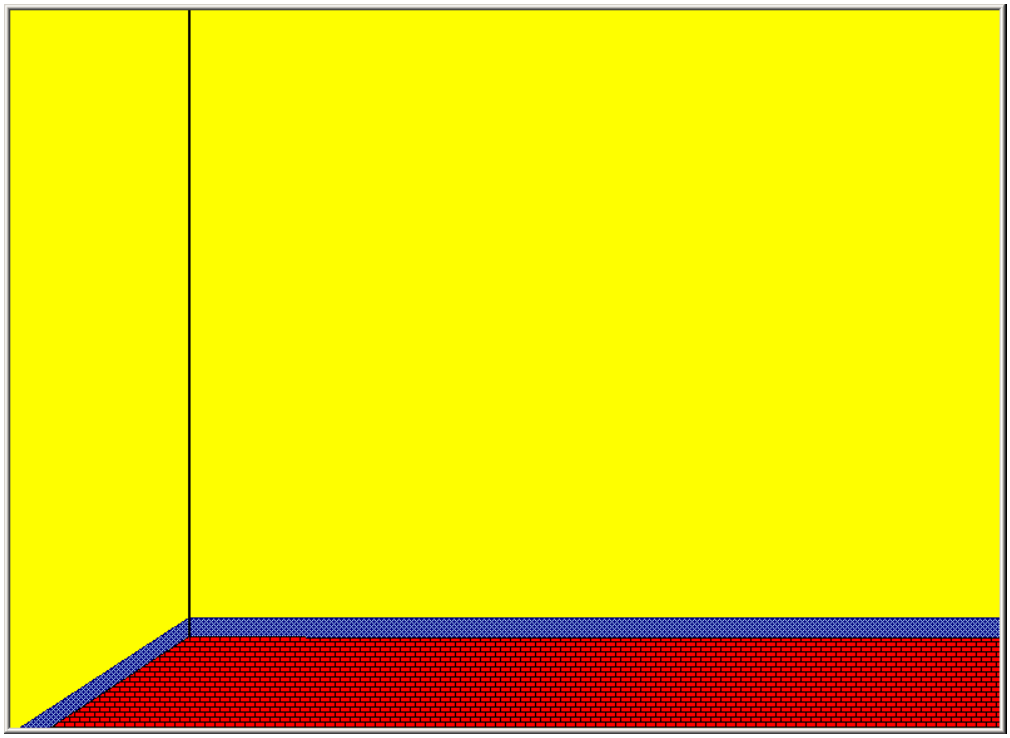
   The dialog should now look as follows:



3. Repeat for Main.pag with the following properties:

   Page Title = **Main**
   Border Style = **THICK**
   Display Mode = **REPLACE**
   Top = **0**
   Left = **0**
   Height = **582**
   Width = **800**
   Background colour = **Yellow**

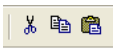
4. Still on **Main.pag**, draw a 3D floor using different graphics objects from the graphics object toolbox. Use a line object, and 2 Polygons. Use the colour Palette to change the fill patterns and colours. The page should now look like this



Now draw the Coffee machine as shown in the picture below

1. Use a **Text object** from the Graphics Toolbar for the display, and another for the 2 digit selection code. Use hash '#' characters to signify space to be replaced with a value.

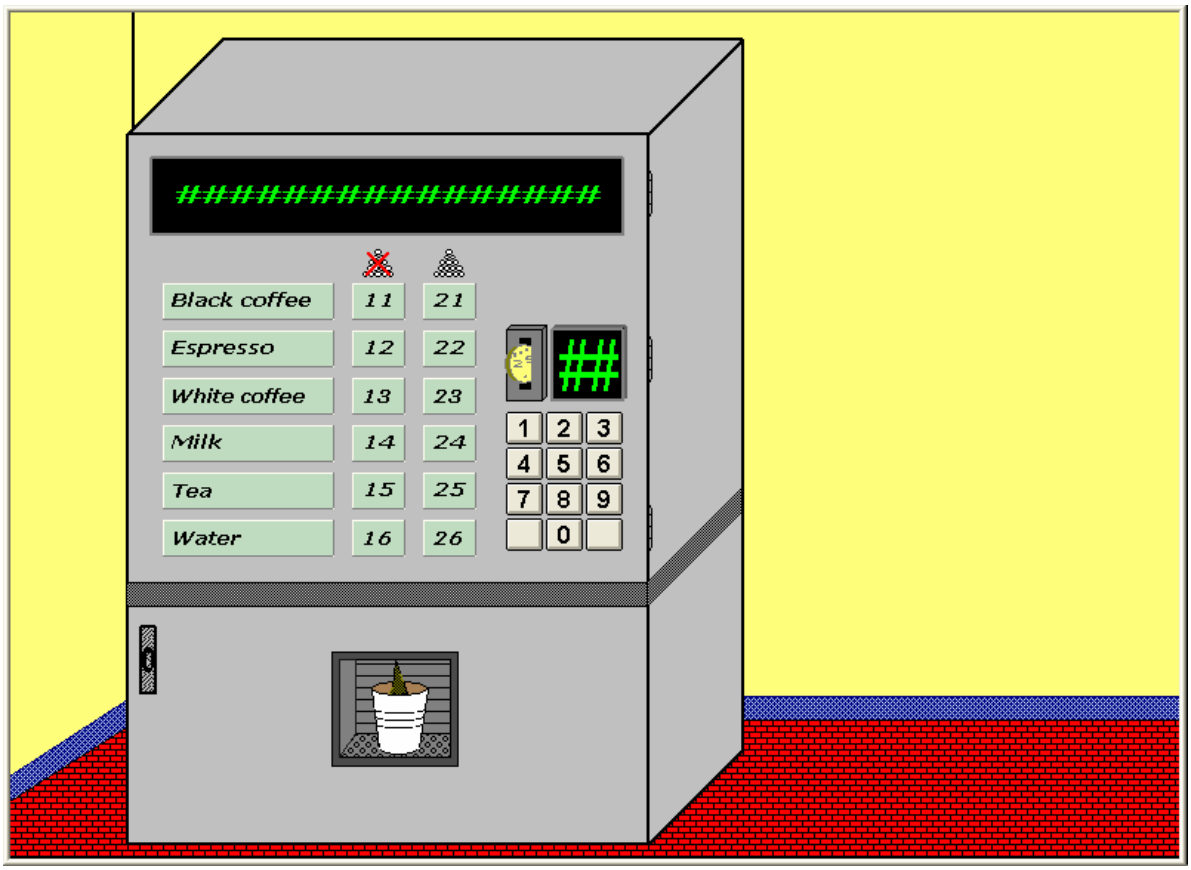
2. Use 10 **Push Buttons** for the user selection buttons, with text 0 to 9 respectively
3. Add a menu showing the selection numbers for the different drinks, both with and without sugar



Tip: You can use Copy and Paste to quickly duplicate similar items

Tip: You can use the Alignment toolbar (or View Menu, Alignment) to quickly align all selected objects horizontally or vertically

The page should now resemble the following:



4. On the upper right corner, include new Push Buttons for displaying different pages for later exercises. Label the buttons **Alarms**, **Graphs & Charts**, **Data Logging**, **File Handling**, **Report Generation**, **Database** and **Security**.

## STEP 04 – GRAPHICS LIBRARY

Now we'll add these to a Graphics Library for later re-use.

***1, 2, 3…***

1. Select the objects used for the background. Holding the Shift key allows selected objects to be added to the selection.
2. Group the objects together by pressing <Ctrl>+G or selecting **Group** from the **Edit** menu, and rename the group object to **Background**
3. Select all the objects in the coffee machine by dragging the selection box around all the objects.
4. Group the objects together. Rename the group object to **CoffeeMachine**

5. Open the Graphics Library Editor window, and position so both can be clearly seen.
6. Add a library called **My Library**
7. Add the grouped objects to your new library by just dragging them from the page to the Library Window. Note that object animations and scripts are copied to the Library.

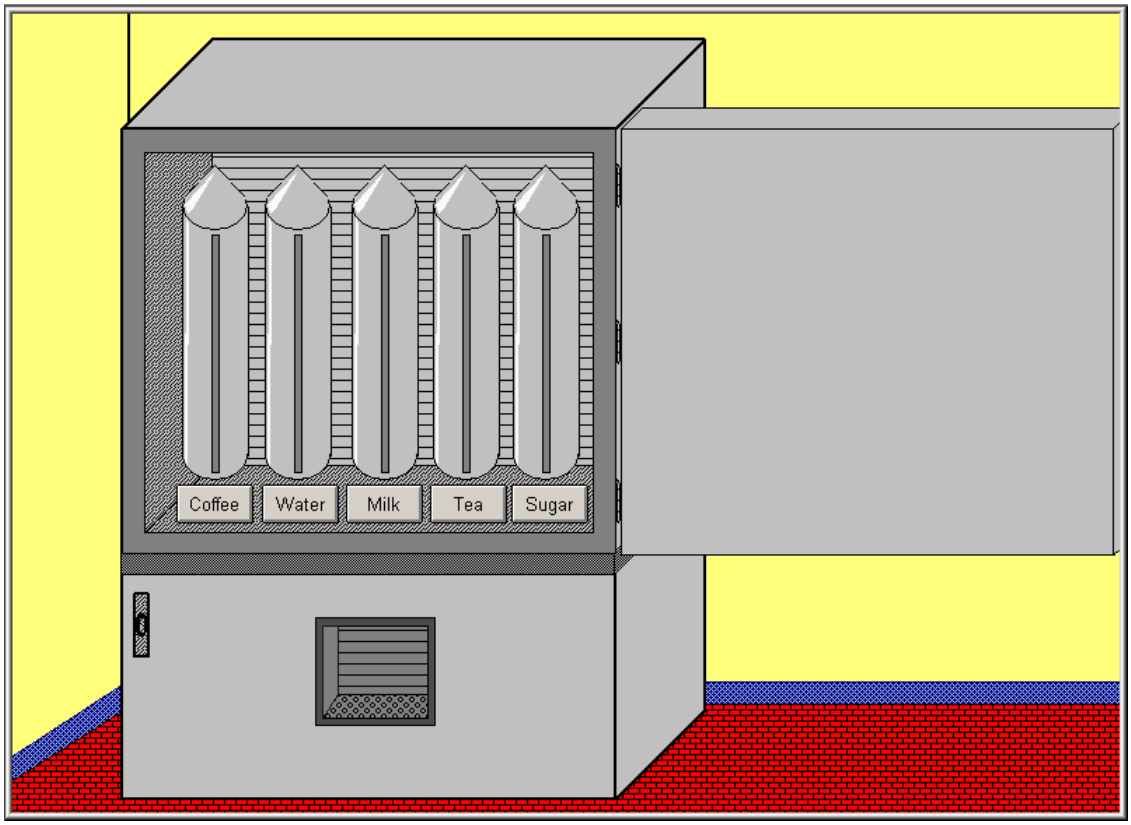
Now we'll re-use these objects.

***1, 2, 3…***

1. Open the page "**open.pag**".

2. Open the Graphics Library Editor window, and position so both can be clearly seen.
3. Drag the Background and Coffee Machine to the new page.
4. Modify the page to show the machine open, as shown below.
   - Delete the menu objects
   - Add the open door
   - Add storage tanks (from the TANKS_16 library)
   - Add Push buttons under the tanks

Tip: You can also use Copy and Paste to quickly create duplicate objects like the tanks and buttons.



# STEP 05 – ANIMATION

We have 2 pages but the application still doesn't actually do anything. Now we'll add the points and animations to make the machine function.

Add the required points.

***1, 2, 3…***

1. Ensure the project is saved before continuing
2. From the toolbar open the **Point Editor** (or Utilities Menu, Point Editor)
3. Add the points required for this step with details as listed in the table below. The Description column is optional as it only helps explain to you the function of the point.

| Point name | Point type | Default Value | Description (optional) |
|---|---|---|---|
| coffee_level | Integer | 2000 | Total level of stored coffee |
| credit | Boolean | | TRUE when coin inserted into the machine |
| displaytext | Text | ********** | Top line of machine display |
| milk_level | Integer | 2000 | Total level of stored milk |
| selection | Integer | | Number of selection made |
| sugar_level | Integer | 2000 | Total level of stored sugar |
| tea_level | Integer | 2000 | Total level of stored tea |
| time_counter | Integer | | Control pouring of drink |
| water_level | Integer | 2000 | Total level of stored water |

Opening and closing the machine:

***1, 2, 3…***

1. Open the page called **open**
2. Select the door lock object on the bottom left
3. From the toolbar, open the **Animation Editor** (or Utilities Menu, animation Editor)
4. Add an animation for **Display Page** and select the page to be opened as **Main**. Note that because the pages were defined as **Replace**, when one page opens it will replace (and close) pages underneath automatically.
5. Repeat for the door lock on the page called **Main** to open the page called **Open**

Inserting the coin:

***1, 2, 3…***

1. Open the page called **Main**
2. Select the coin and add an animation for **Execute Script** with the script code "credit = 1" to acknowledge that money has been entered.
3. Also add an animation for **Visibility** controlled by the value of **credit**. Remember the coin should be visible while **credit** is false, or should be invisible when **credit** is true or "credit == 1"

Updating displays, and pouring coffee:

***1, 2, 3…***

1. Select the main display object
2. Add a **Display Value (Text)** animation to with an expression **displaytext**. Note: if this animation is not available then the object may not have been originally created as a Text Object type.
3. Add a **Display Value (Analogue)** animation to the 2 digit display with an expression **selection**
4. Add a **Resize (Height)** animation to coffee liquid object with an expression **time_counter** to simulate the pouring. Minimum and Maximum should be **0** and **50** respectively.

Displaying storage levels:

***1, 2, 3…***

1. Open the page called **open**
2. Select the storage tank for the coffee. Use the Workspace or Ungroup to find the rectangular window and add a **Percentage Fill (Vertical)** animation to the coffee storage tank viewing window with an expression **coffee_level** and minimum and maximum of **0** and **2000** respectively
3. Repeat for the other storage tanks with the appropriate points for the expressions.
4. Add an **Edit Value (Analogue)** animation to the button below the coffee tank with an expression **coffee_level**
5. Repeat for each button using appropriate point for each level

Now would be a good time to run the application by clicking the toolbar button "Run the project" (or Project Menu, Run) and test that you can open and close the machine, and add credit to the machine.

## STEP 06 – SCRIPTS

To extend the functionality we will add a **Reset()** subroutine for initializing, make the coffee liquid and the cup invisible and set a text such as "Insert coin" for the coffee machine display. This assumes the cup object, pouring coffee and main display objects are called **cup**, **coffee** and **Text_2** respectively. Either adjust the script for your object names, or rename them in the Workspace to these values.

Add the points required by this step.

***1, 2, 3…***

1. Open the Point Editor
2. Add the points with details as listed in the table below. The Description column is optional as it only helps explain to you the function of the point.

| Point name | Point type | Default Value | Description (optional) |
|---|---|---|---|
| pressed | Boolean | | TRUE once first digit has been pressed |
| ready | Boolean | | Machine is ready to make a drink |
| selection_made | Boolean | | TRUE when a selection has been made |

***1, 2, 3…***

1. Open the Workspace editor
2. Right click on Project Script to add a new **Project Script**
3. Set Trigger Event of **Subroutine** named **Reset** with the following script code to reset variables:

```
credit = 0
selection = 0
ready = TRUE
pressed = false
selection_made = false
main.coffee.visible(0)
main.cup.visible(0)
displaytext = "Insert Coin"
main.Text_2.blink( black )
```

4. On **Main.pag** add a Page Script with a Trigger Event of **On Initialisation** to execute every time the page is loaded with the following code:

```
CALL Reset( )
```

When the coin is inserted, we want to make a sound and change the displayed command.

***1, 2, 3…***

1. Copy the sound file **cashreg.wav** from the folder of the sample programs to the project folder **C:\Coffee**
2. Create a Page Script with Trigger Event of **On Condition** with an Expression of **credit == 1** and script code of :

```
displaytext = "Make a Selection"
PlaySound("c:\coffee\cashreg.wav")
```

Include a script for displaying the user selection in the Selection Display and assign this selection to the **selection** variable. Use the variables **selection** to store the number typed, **pressed** to note when the first digit has been entered and **selection_made** to determine when the second digit has been entered therefore the selection is complete.

***1, 2, 3…*** 1. Add a **Execute Script** animation to the Push Button with caption "1" with the following code:

```
IF pressed == TRUE THEN
        selection = (selection * 10 ) + 1
        pressed = FALSE
        selection_made = TRUE
ELSE
        selection = 1
        pressed = TRUE
        selection_made = FALSE
ENDIF
```

2. Repeat for the other buttons, replacing both the 1's with the appropriate digit.

To round off this step, include a script to produce the selection entered by the user. This script should:

- Display a message for informing the user what is happening
- Display the cup and the coffee liquid.
- Play a sound file to simulate the coffee dropping, and one when finished
- Display a message to notify the user to pick up the coffee

***1, 2, 3…*** 1. Copy the sound files **glu.wav** and **finished.wav** from the folder of the sample programs to the project folder **C:\Coffee**

2. Add a page script to trigger **On Condition** called **Make the Drink** with an expression of "**(selection_made == TRUE) and (credit == 1) and (ready == TRUE)**". Add the following code:

```
ready = FALSE
displaytext = "Making your selection"
cup.visible( 1)
coffee.visible (1)
for time_counter = 0 to 50
        'just looping for coffee animation
next
PlaySound("c:\coffee\glu.wav")
PlaySound("c:\coffee\glu.wav")
PlaySound("c:\coffee\glu.wav")
PlaySound("c:\coffee\glu.wav")
Sleep(2000)
coffee.visible(0)
PlaySound("c:\coffee\finished.wav")
displaytext = "Pick up your selection"
```

3. Add an **Execute Script** animation to coffee cup with code to call the **Reset** subroutine when the cup is clicked.

Now's a good time to run the application and test the changes so far.

# STEP 07 – RECIPES

So we can buy, select and make a drink, but how do we handle making different types of drinks? By creating a 'Recipe' for each choice, predetermined amounts of ingredients can be used.

Add the points required by this step.

| Point name | Point type | Default Value | Description (optional) |
|---|---|---|---|
| aux_text | Text | | Temporary text area for conversions |
| coffee_amount | Integer | | Amount of coffee for this selection |
| milk_amount | Integer | | Amount of milk for this selection |
| selection_name | Text | | Name of selection made |
| sugar_amount | Integer | | Amount of sugar for this selection |
| tea_amount | Integer | | Amount of tea for this selection |
| water_amount | Integer | | Amount of water for this selection |

***1, 2, 3…***

1. Open the Recipe Editor.
2. Create a new recipe called "**11**" and add the following ingredients. Note that for simplicity the Recipe name is made the same as the selection code.

| | Ingredient Name | Point | Quantity |
|---|---|---|---|
| **Ingredient 1** | Name | selection_name | Black Coffee |
| **Ingredient 2** | Coffee | coffee_amount | 100 |
| **Ingredient 3** | Water | water_amount | 100 |
| **Ingredient 4** | Sugar | sugar_amount | 0 |
| **Ingredient 5** | Milk | milk_amount | 0 |
| **Ingredient 6** | Tea | tea_amount | 0 |

3. Create remaining recipes with ingredient quantities as detailed in the list below.

Tip: You can Copy and Paste Recipe 11 and the recipe name will automatically be incremented. The Ingredient values may then just be edited.

| Recipe Name | Name | Coffee | Water | Sugar | Milk | Tea |
|---|---|---|---|---|---|---|
| 12 | Espresso | 50 | 75 | 0 | 0 | 0 |
| 13 | White Coffee | 50 | 50 | 0 | 50 | 0 |
| 14 | Milk | 0 | 0 | 0 | 100 | 0 |
| 15 | Tea | 0 | 100 | 0 | 0 | 100 |
| 16 | Water | 0 | 100 | 0 | 0 | 0 |

4. Copy all the previous recipes for the remaining selections (21, 22, 23, 24, 25, 26) and change the sugar ingredient to **50**, and name with a "**with Sugar**" suffix.
5. Add the code to the **Make the Drink** script to download the recipe ingredients :

```
aux_text = ValueToText( selection )
DownloadRecipe (aux_text)
Sleep( 1000)
'Take quantities from storage
coffee_level = coffee_level - coffee_amount
water_level = water_level - water_amount
milk_level = milk_level - milk_amount
tea_level = tea_level - tea_amount
sugar_level = sugar_level - sugar_amount
```

When run, you will see that each time a drink is made the storage tanks are emptied by the correct amount for each ingredient.

# STEP 08 – ALARMS

We can provide the mechanism to create an alarm should any of the resources run out.

***1, 2, 3…***

1. Open the Alarm Editor.
2. Add the alarms with details from the list below.

| Name | Expression | Raised Message | Normal Message |
|---|---|---|---|
| coffee_level_alarm | coffee_level <= 0 | Coffee deposit is empty | Coffee deposit is refilled |
| water_level_alarm | water_level <= 0 | Water deposit is empty | Water deposit is refilled |
| milk_level_alarm | milk_level <= 0 | Milk deposit is empty | Milk deposit is refilled |
| tea_level_alarm | tea_level < = 0 | Tea deposit is empty | Tea deposit is refilled |
| sugar_level_alarm | sugar_level <= 0 | Sugar deposit is empty | Sugar deposit is refilled |

Test the alarms to see them in action.

Add Alarm viewing page:

***1, 2, 3…***

1. Make a new page with the following properties and save it to the project folder.
   Page Title = **Alarm**
   Border Style = **THICK**
   Display Mode = **POPUP**
   Top = **0**
   Left = **505**
   Height = **582**
   Width = **295**
2. Use the Workspace to remove the **Display On Run** option from the popup menu.
3. Add an 'Alarm' Graphical Object for displaying alarms in real time. Configure the object as you wish.
4. Add a button with text "Alarm Status Viewer" and add script code:

   DisplayAlarmStatus()
5. Add a button with text "Alarm History Log" and add script code:

   DisplayAlarmHistory()
6. Add a 'Close' button with an animation to close the page.
7. Open **Main** page and add an animation to the "Alarms" button to display the new page called **Alarm**.

This concludes the section on alarms. Test out your application then move on to the next step.

# STEP 9 – GRAPHS AND CHARTS

Real-time data can be graphically represented on Bar Charts and trend graphs.

Add the points required by this step.

| Point name | Point type | Default Value | Description (optional) |
|---|---|---|---|
| milk_temperature | Integer | 70 | Temperature of milk (degrees C) |
| water_temperature | Integer | 80 | Temperature of water (degrees C) |

***1, 2, 3…***

1. Make a new page with the following properties and save it to the project folder.
   Page Title = **Graphs**
   Border Style = **THICK**
   Display Mode = **POPUP**
   Top = **0**
   Left = **505**
   Height = **582**
   Width = **295**
2. Add a 'Chart' Graphical Object for displaying alarms in real time. Configure the object to graph the 5 resource levels (coffee_level etc), with appropriate labels
3. Use the Workspace to remove the **Display On Run** option from the popup menu.
4. Add a 'Trend Graph' Graphical Object for displaying **water_temperature** and **milk_temperature**. Configure as preferred, scaling from 0 to 200.
5. Add 2 sliders and link to **water_temperature** and **milk_temperature** to simulate changing temperatures.
6. Add a 'Close' button with an animation to close the page.
7. Open **Main** page and add an animation to the "Graphs & Charts" button to display the new page called **Graphs**.

# STEP 10 – DATA LOGGING

In addition to simple trending, we can add comprehensive Data Logging. First create the items to log.

***1, 2, 3…***

1. Open the Workspace Editor and select the **Data Logging** tab
2. Right click and select **Add DataSet…**
3. Right click on the added Data Set and select **Add Item…**. Add the item **Milk Temperature** and enter **milk_temperature** in the expression field.
4. Repeat to add an item called **Water Temperature** for point **water_temperature**.

Then create a page to view and export the data and control logging:

***1, 2, 3…***

1. Make a new page with the following properties

   Page Title = **Datalog**
   Border Style = **THICK**
   Display Mode = **POPUP**
   Top = **0**
   Left = **505**
   Height = **582**
   Width = **295**
2. Use the Workspace to remove the **Display On Run** option from the popup menu.
3. Add a button to automatically open the data log viewer using the command "**OpenLogView("Dataset1", "Milk Temperature,Water Temperature", "")**"
4. Add a button to export data to CSV using the command "**ExportLog("Dataset1", "Milk Temperature,Water Temperature", "CSV", 0, "c:\coffee\export.txt")**"
5. Add a button to manually restart logging using the command "**StartLogging("Dataset1")**"
6. Add a button to manually restart logging using the command "**StopLogging("Dataset1")**"
7. Add a 'Close' button with an animation to close the page.
8. Open **Main** page and add an animation to the "Data Logging" button to display the new page called **Datalog**.

# STEP 11 – FILE HANDLING

We'll add a file handling page allowing the selected drink to be written to a file on disk. The file format will look like this:

03:42:52 10/30/2002 – User Selection: 14
03:44:28 10/30/2002 – User Selection: 22
03:53:28 10/31/2002 – User Selection: 16

First add the points required by this step.

| Point name | Point type | Default Value | Description (optional) |
|---|---|---|---|
| aux_text2 | Text | | Temporary text area for conversions |
| fileindex | Integer | | Index to current position in text file |
| filename | Text | | Name of text file to read from. |
| ret | Integer | | General purpose 'return' value from various script functions |

Then write the subroutine to do the work. This subroutine should be called every time a drink is made.

***1, 2, 3…***

1. At the Project level, add a **subroutine** script called **WriteToDisk**
2. Add an integer parameter called **sel**
3. Add code to format the data and write to disk:

```
aux_text2 = ValueToText( selection )
aux_text = $Time + " "+ $Date+ " - User selection: "+aux_text2
ret = WriteMessage("selections.txt", -1, aux_text, true)
```

4. Add script code to **Make the Drink** script to call the new subroutine

Add the detail to the File Handling page:

***1, 2, 3…***

1. Make a new page with the following properties

   Page Title = **File**
   Border Style = **THICK**
   Display Mode = **POPUP**
   Top = **0**
   Left = **505**
   Height = **582**
   Width = **295**

2. Use the Workspace to remove the **Display On Run** option from the popup menu.
3. Add a button with caption **Select File** with the following script for selecting the previously created file (selections.txt):

   ```
   filename = SelectFile("Text Files (*.txt)|*.txt|All Files (*.*)|*.*||", "C:\Coffee")
   ```

4. Add a button with caption **Line Read** with the following script for reading a line of text from the selected file:

   ```
   aux_text = ""
   ReadMessage(filename, fileindex, aux_text, 42)
   fileindex = fileindex + 42
   ```

5. Add a button with caption **Reset Index** with the following script for resetting the cursor to the start of the file:

   ```
   fileindex = 0
   ```

6. Add 3 Text objects with Display animations to display the points **filename**, **aux_text**, and **fileindex**.
7. Add a ‘Close’ button with an animation to close the page.
8. Open **Main** page and add an animation to the “File Handling” button to display the new page called **File**.

# STEP 12 – REPORT GENERATION

Data can easily be formatted and exported in a textual report, for example in .TXT, .RTF or .HTML format. The template file will contain the following text:

```
                    DEPOSIT LEVEL REPORT


(("DATE: %s", $Date ))
```

(("TIME: %s", $Time ))

(("Coffee Level: %d", coffee_level))
(("Water  Level: %d", water_level))
(("Milk   Level: %d", milk_level))
(("Tea    Level: %d", tea_level))
(("Sugar  Level: %d", sugar_level))

(("Active Alarms: %d", $ActiveAlarms))

***1, 2, 3…***

1. Create a file in the project directory called **source.txt** with the template text. This template will be evaluated and the values within brackets formatted at runtime.
2. Make a new page with the following properties

   Page Title = **Report**
   Border Style = **THICK**
   Display Mode = **POPUP**
   Top = **0**
   Left = **505**
   Height = **582**
   Width = **295**
3. Use the Workspace to remove the **Display On Run** option from the popup menu.
4. Add a button with caption **Generate Report** with the following script for creating the output report file (report.txt):

   GenerateReport("C:\coffee\source.txt","report.txt")
5. Add a button with caption **View Report** with the following script for launching an associated viewer:

   ViewReport("C:\coffee\report.txt")
6. Add a button with caption **Edit Report** with the following script for loading the template into an installed editor:

   EditFile("C:\coffee\source.txt")
7. Add a ‘Close’ button with an animation to close the page.
8. Open **Main** page and add an animation to the “Report Generation” button to display the new page called **Report**.

Creating HTML reports:

***1, 2, 3…***

1. Create a file in the project directory called **source.htm** with the template text. If using Word or more powerful html editor, feel free to experiment with different fonts, font sizes and colours etc.
2. Add 3 buttons to Generate, View and Edit the report as per the text example.

# STEP 13 – DATABASE ACCESS

We will add the facility to write the ingredients used to a standard database, and add a page to read through records previously written. First we must create the database template using Microsoft Access. Please consult your Microsoft documentation for full details of these operations. If you do not have Microsoft Access, you can still read and write Access files from CX-Supervisor but you will need to copy a database file from the demo folder to use as a template. Alternatively you can use any other ADO compatible data source, with suitable changes to the connection details.

Add the point required by this step.

| Point name | Point type | Default Value | Description (optional) |
|---|---|---|---|
| record_index | Integer | | Index to current database record |

***1, 2, 3…***

1. Launch Microsoft Access and create a new database called **coffee.mdb** in the project directory **C:\Coffee**
2. Create a new table, in Design View if prompted.
3. Insert 6 new Fields and rename them as per the table below.
4. Open the table Design, and configure the data type and field size from the table below.
5. Save the table as **Ingredients**, without an index if prompted
6. Close Microsoft Access, saving any changes.

| Field Name | Field Data Type | Field Size |
|---|---|---|
| Selection | Memo | - |

| | | |
|---|---|---|
| Coffee | Number | Integer |
| Water | Number | Integer |
| Milk | Number | Integer |
| Tea | Number | Integer |
| Sugar | Number | Integer |

You should now have a template Access database (or other) to use. The next step is to add the database connections in the Developer.

***1, 2, 3…***

1. Open CX-Supervisor Developer and load the project.
2. Open the Workspace and switch to the Database tab
3. Right click and add a new connection called **Connection1** and specify the new template file **C:\Coffee\coffee.mdb** as the Data Source.
4. Right click the Connection and select **Connect**. The icon will change to indicate the connection is live.
5. Add a new Recordset for adding records, called **Recordset1** to the connection and ensure **Ingredients** is selected as the table name for the Source. To allow writing to the database make sure **Pessimistic** (or **Optimistic**) lock is selected.
6. Add a Field called **selection** linked to point **selection_made** and Field **Selection**. Change the Field Property to **Add**.
7. Repeat to add **coffee** linked to point **coffee_amount** and Field **coffee** with the Field Property **Add** and repeat for **milk**, **sugar**, **tea**, and **water**.
8. Add a new Recordset for reading records, called **Recordset2** to the connection and ensure **Ingredients** is selected as the table name for the Source.
9. Copy each field from Recordset1 and Paste to Recordset2. Change all the Field Properties from **Add** to **Value**.

With the template of the database now defined, it is very to easy write the data to the database

***1, 2, 3…***

1. At the Project level, add a **subroutine** script called **WriteToDatabase**
2. Add code to write to the database, requery the table so that the latest data can be seen, and navigate back to the original record:

```
DBAddNew( "Connection1.Recordset1" )
DBExecute( "Connection1.Recordset2", "Requery" )
DBMove( "Connection1.Recordset2", "Position", record_index)
```

3. Add script code to **Make the Drink** script to call the new subroutine

Written database records can now be viewed:

***1, 2, 3…***

1. At the Project level, add a **subroutine** script called **ReadFromDatabase**
2. Add code to read to the database:

```
DBRead( "Connection1.Recordset2" )
record_index = DBProperty( "Connection1.Recordset2",
"CurrentRecord" )
```

3. Make a new page with the following properties

Page Title = **Database**
Border Style = **THICK**
Display Mode = **POPUP**
Top = **0**
Left = **505**
Height = **582**
Width = **295**

4. Use the Workspace to remove the **Display On Run** option from the popup menu.
5. Add a button for navigating to the next record with the script:

```
ret = DBProperty( "Connection1.Recordset2", "RecordCount" )
IF record_index < ret THEN
        DBMove( "Connection1.Recordset2", "Next" )
ENDIF
CALL ReadFromDatabase( )
```

6. Add a button for navigating to the previous record with the script:

```
DBMove( "Connection1.Recordset2", "Previous" )
CALL ReadFromDatabase( )
```

7. Add a Text object with a Display animation to display the point **record_index**.

8. Add 6 Text objects with Display animations to display the points **selection_made**, **coffee_amount**, **water_amount**, **milk_amount**, **tea_amount**, and **sugar_amount**.
9. Add a 'Close' button with an animation to close the page.
10. Open **Main** page and add an animation to the "Database" button to display the new page called **Database**.

# STEP 14 – SECURITY

Operations can be restricted to users with acceptable security privileges. We will only allow Supervisors or those with higher privileges to open the machine.

***1, 2, 3…***

1. Open the **Main** page
2. Select the door lock on the bottom left of the machine.
3. Open the Animation Editor and select the Display Page action previously added. On the toolbar, change the Security Level for this operation from **All Users** to **Supervisor**.
4. Add an **Execute Script** animation with the following code:

```
IF $SecurityLevel < 2 THEN
        Message( "You are not authorized to open the machine")
ENDIF
```

And add a page to control the security

***1, 2, 3…*** 1. Make a new page with the following properties

Page Title = **Security**
Border Style = **THICK**
Display Mode = **POPUP**
Top = **0**
Left = **505**
Height = **582**
Width = **295**

2. Use the Workspace to remove the **Display On Run** option from the popup menu.
3. Add 2 buttons with captions **Login** and **Logout** and add code to call script functions (just "Login()" and "Logout()" respectively).
4. If you desire, add Text objects with Display animations to display the points **$SecurityLevel**, **$SecurityName**, and **$UserName**.
5. Add a 'Close' button with an animation to close the page.
6. Open **Main** page and add an animation to the "Security" button to display the new page called **Security**.

You can now test the security. Note the default user names and passwords below are case sensitive:

| **User** | **Password** |
|---|---|
| Operator | Operator |
| Supervisor | Supervisor |
| Manager | Manager |
| Designer | Designer |

This concludes the Advanced tutorial where you have learnt about Graphics Libraries, Recipes, more animations, different scripts, Alarms, Graphs and Charts, Data Logging, File handling, Report generation, Database connectivity, and Security. These topics have only been touched on briefly, to show some basic possibilities and are further detailed in the User Manual and Script Reference manual.

The final solution will have been installed and can be run from the Start button, in **Demos** under the CX-Supervisor folder.

# GLOSSARY

**ADO**
ADO stands for ActiveX Data Objects and is data access technology which uses OLE-DB to access data sources in a uniform way e.g. MS-Access databases, MS-Excel spreadsheets and Comma Separated Variable files.

**Application**
A software program that accomplishes a specific task. Examples of applications are CX-Supervisor, CX-Programmer, Microsoft Word for Windows and Microsoft Excel. CX-Supervisor and its development environment allows the creation and testing of new applications through a Graphical User Interface (GUI).

**Bitmap**
The representation of an image stored in a computer's memory. Each picture element (pixel) is represented by bits stored in the memory. In CX-Supervisor a bitmap image can be installed as a single object.

**Communications Driver**
The relevant communications management system for OMRON PLCs in conjunction with Microsoft Windows, providing facilities for other CX Automation Suite software to maintain PLC device and address information and to communicate with OMRON PLCs and their supported network types.

**CX-Supervisor**
A software application which creates and maintains graphical user interfaces and communicates with **PLCs** and other I/O mechanisms.

**DDE**
Dynamic Data Exchange. An obsolete Microsoft technology which provides a channel through which correctly prepared programs can actively exchange data and control other applications within Microsoft Windows.

**Development environment**
Applications are created using the development environment within CX-Supervisor. On completion, the finished application can be delivered as a final customer application to be run by the run-time environment.

**Dongle**
A hardware key that plugs in the USB port that unlocks operation.

**GUI**
Graphical User Interface. Part of a program that interacts with the user and takes full advantage of the graphics displays of computers. A GUI employs pull-down menus and dialog boxes for ease of use. Like all Microsoft Windows based applications, CX-Supervisor has a GUI.

**I/O type**
Input / Output type. An attribute of a point that defines the origin and destination of the data for that point. The data for a point can originate (be *input* from) and is destined (is *output* to) to the internal computer memory, PLC, DDE target application.

**Icon**

Pictorial representations of computer resources and functions. The CX-Supervisor development environment and run-time environment are run from icons. Icons are also used in CX-Supervisor to indicate an OLE object.

**Microsoft Excel**

A spread sheet application.

**Microsoft Windows**

A windowing environment for personal computers, that is noted for its GUI, and for features such as multiple typefaces, desk accessories (such as a clock, calculator, calendar and notepad), and the capability of moving text and graphics from one application to another via a clipboard.

CX-Supervisor will run only under Microsoft Windows.

**Microsoft Word**

A word processing application.

See also SVGA mode and VGA mode.

**Object**

In CX-Supervisor, an object can be text, graphics, a control, a bitmap, or OLE object as created in the development environment. A complex object can exist as a combination of two or more objects of any of the above types. Specifically, graphical objects can be categorised as a line, an arc, a polygon (including a square and rectangle), a round rectangle, an ellipse (including a circle), or a Polyline. A control is essentially a complex graphic object and is specifically either a button, a toggle button, a slider, a trend graph, a rotational gauge or a linear gauge.

**OLE**

Object Linking and Embedding. Used to transfer and share information between Microsoft Windows based applications and accessories. When OLE is used in CX-Supervisor, it is possible to view or even edit a file from a target application.

**OLE-DB**

OLE-DB is the underlying database technology, on which ADO relies. OLE-BD is designed to be the successor to ODBC.

**Operator**

A symbol used as a function, with infix syntax if it has two arguments (e.g. "+") or prefix syntax if it has only one argument (e.g. NOT). The CX-Supervisor script language uses operators for built-in functions such as arithmetic and logic.

**Pages**

The combination and manipulation of pages containing objects within projects forms the basis of CX-Supervisor. More than one page can exist for each project. The pages in a project provide the visual aspect of CX-Supervisor corresponding to a display with the objects contained in each page providing a graphical representation of the system being monitored.

**Pixel** A single displayable point on the screen from which a displayed image is constructed. The screen resolution of the computer's Visual Display Unit (VDU) is defined by the number of pixels across and the number of pixels down (e.g. 1024 x 768).

**PLC** Programmable Logic Controller.

**Point** A point is used to hold a value of a predefined type - Boolean, Integer, Text, etc. The contents of a point may be controlled by an object or I/O mechanism such as DDE. The contents of a point may control the action or appearance of an object, or be used for output via an I/O mechanism.

**Project** A CX-Supervisor application will consist of one or a number of pages linked together. The pages may contain passive or active graphics, text or animations, and may be grouped together logically to form a project. A project may consist of many pages, or simply a single page. Projects may be built and tested within the CX-Supervisor development environment, and run stand-alone under the CX-Supervisor run-time environment.

Only one project at a time may be open for editing within the CX-Supervisor development environment.

**Run Time Environment** Applications are run using the run-time environment of CX-Supervisor, following creation of the application in the CX-Supervisor development environment.

**SCADA** Supervisory Control and Data Acquisition.

**SVGA mode** A mode of video display that provides 800 × 600 pixel resolution (or higher) with 16 or more colours and is supported on Super Video Graphics Adapter systems.

**Topic** Within the CX-Supervisor script language, Topic is used in DDE functions to specify a file name pertaining to an outside application. Using DDE functions, CX-Supervisor allows the opening of a file, part of the server application.

**VGA mode** A mode of video display that provides 640 × 480 pixel resolution with 16 colours and is supported on Video Graphics Adapter systems.

**Windows Desktop** An integral part of Microsoft Windows which allows Microsoft Windows based applications to be started from icons and for all applications to be organised. CX-Supervisor can be run from Windows Desktop.

INDEX


## *A*

About this Manual · 2
Accessing the Help System · 9
Activating a Dongle · 6
Activating CX-Supervisor Token · 6
ActiveX Data Objects · 49
Adding a Point - *Tutorial*
Advanced - *Tutorial*
Animation - *Tutorial*
Animations · 16–20
Application · 49

## *B*

Bitmaps · 49

## *C*

Communications Driver · 49
Concepts of Projects - *Tutorial*
Copy Protection · 4
  Backing up Token Diskettes · 8
  Disk Cache Operation · 7
  Disk Compression · 7
  Disk Defragmentors · 8
  File Maintenance · 7
  Protection using a Token · 4
  Restrictions · 7
  System Backup · *See*
  Token Installation · 7
Customising Settings · 9
CX-Supervisor · 49
CX-Supervisor Pages - *Tutorial*

## *D*

Development Environment · 49
Dongle · 5
  Activating · 6
  Fitting · 5
Dynamic Data Exchange · 49

## *F*

Fitting a Dongle · 5

## *G*

Getting Started · 1, 11
Graphical User Interface · 49
Graphics Editor · 14–16
  View · 12

## *H*

Hardware Communication Interface · 4
Hardware Copy Protection · *See* Dongle. *See* Dongle
Hardware Requirements · 3
Help System · 9
HTML reports · 44

## *I*

Icon · 50
Input / Output Types · 49
Installing CX-Supervisor · 4
Interfaces to Hardware · 4
Introduction to CX-Supervisor · 1, 11

## *M*

Microsoft Excel · 50
Microsoft Windows · 50

# *O*

Object · 50
Object Linking Embedding (OLE) · 50
Operating Systems and Environments · 3
Operator · 50

# *P*

Pages · 50
Pixel · 51
Point · 51
Point Editor
  Create point · 20
Project · 51
  Concepts · 12–13
  Creating a new project · 12

# *R*

Refining Projects- *Tutorial*
Run Time Environment · 51
Runtime
  Testing · 11, 20, 21, 23

# *S*

SCADA · 51
Settings - Customising CX-Supervisor · 9
Simple Tutorial · 12–22
Software Copy Protection · 4
  Using a Token · 6
Starting CX-Supervisor · 1, 8, 11
System Requirements · 3
  Hardware Requirements · 3
  Interfaces to Hardware · 4
  Operating System · 3

# *T*

Testing the Project - *Tutorial*
The First Steps with CX-Supervisor · 11
Token
  Activating a Token · 6
  Backing up Token Diskettes · 8
  Copy Protection · 6
  Disk Cache Operation · 7
  Disk Compression · 7
  Disk Defragmenters · 8
  File Maintenance · 7
  Installing a Token · 7
  Restrictions · 7
  System Backup · 7
Topic · 51
Tutorial
  Adding a Point · 16, 17
  Advanced · 25
  Alarms · 37
  Animation · 18, 31
  Charts · 38
  Concepts of Projects · 12
  CX-Supervisor Pages · 13
  Data Logging · 39
  Database access · 44
  File Handling · 40
  Graphics Editor · 14
  Graphics Library · 29
  Graphs · 38
  HTML Reports · 42
  Reading Files · 40
  Recipes · 36
  Refining the Project · 21
  Report Generation · 42
  Scripts · 33
  Security · 47
  Simple · 12–22
  Testing the Project · 20

# *U*

Using a Token for copy protection · 6
Using the Graphics Editor- *Tutorial*

# *V*

VGA Mode · 51

*W*

Welcome to CX-Supervisor · 1
Windows Desktop · 51